RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23221
PYTHON PROGRAMMING LAB

Laboratory Observation Note Book

Name : AAFRIN FATHIMA N

Year / Branch / Section : 1/B Tech CSBS/A

Register No. : 2116231401001

Semester : 2

Academic Year : 2023-2024

INDEX

Reg. No. : 231401001 Name : AAFRIN FATHIMA N

Year : 1 Branch : CSBS-A Sec : A

01 - Introduction to Python-Variables-Datatypes Input/Output-Formatting

Ex. No. : 1.1 **Date: 14/02/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Converting Input Strings

Write a program to convert strings to an integer and float and display its type.

Sample Input:

10

10.9

Sample Output:

10,<class 'int'>

10.9,<class 'float'>

For example:

Input	Result
10 10.9	10,<class 'int'> 10.9,<class 'float'>

Program:

```
a=int(input())

b=float(input())

print(a,type(a),sep=",")

print(round(b,1),type(b),sep=",")
```

	Input	Expected	Got	
✔	10 10.9	10,<class 'int'> 10.9,<class 'float'>	10,<class 'int'> 10.9,<class 'float'>	✔
✔	12 12.5	12,<class 'int'> 12.5,<class 'float'>	12,<class 'int'> 12.5,<class 'float'>	✔
✔	89 7.56	89,<class 'int'> 7.6,<class 'float'>	89,<class 'int'> 7.6,<class 'float'>	✔
✔	55000 56.2	55000,<class 'int'> 56.2,<class 'float'>	55000,<class 'int'> 56.2,<class 'float'>	✔
✔	2541 2541.679	2541,<class 'int'> 2541.7,<class 'float'>	2541,<class 'int'> 2541.7,<class 'float'>	✔

Ex. No. : 1.2 **Date: 14/02/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Gross Salary

Ramesh's basic salary is input through the keyboard. His dearness allowance is 40% of his basic salary, and his house rent allowance is 20% of his basic salary. Write a program to calculate his gross salary.

Sample Input:

10000

Sample Output:

16000

For example:

Input	Result
10000	16000

Program:

```
sal=int(input())

da=sal*40/100

hra=sal*20/100

gs=sal+da+hra

print(round(gs))
```

	Input	Expected	Got	
✔	10000	16000	16000	✔
✔	20000	32000	32000	✔
✔	28000	44800	44800	✔
✔	5000	8000	8000	✔

Ex. No. : 1.3 **Date: 14/02/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Square Root

Write a simple python program to find the square root of a given floating point number. The output should be displayed with 3 decimal places.

Sample Input:

8.00

Sample Output:

2.828

For example:

Input	Result
14.00	3.742

Program:

```
import math

a=float(input())

b=math.sqrt(a)

print(format(b,'.3f'))
```

	Input	Expected	Got	
✔	8.00	2.828	2.828	✔
✔	14.00	3.742	3.742	✔
✔	4.00	2.000	2.000	✔
✔	487	22.068	22.068	✔

Ex. No. : 1.4 **Date: 14/02/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Gain percent

Alfred buys an old scooter for Rs. X and spends Rs. Y on its repairs. If he sells the scooter for Rs. Z (Z>X+Y). Write a program to help Alfred to find his gain percent. Get all the above-mentioned values through the keyboard and find the gain percent.

Input Format:

The first line contains the Rs X

The second line contains Rs Y

The third line contains Rs Z

Sample Input:

10000

250

15000

Sample Output:

46.34 is the gain percent.

For example:

Input	Result
45500 500 60000	30.43 is the gain percent.

Program:

```
x=int(input())
y=int(input())
z=int(input())
gain=(z-x-y)*100/(x+y)
print(format(gain,'.2f'),"is the gain percent.")
```

	Input	Expected	Got	
✔	10000 250 15000	46.34 is the gain percent.	46.34 is the gain percent.	✔
✔	45500 500 60000	30.43 is the gain percent.	30.43 is the gain percent.	✔
✔	5000 0 7000	40.00 is the gain percent.	40.00 is the gain percent.	✔
✔	12500 5000 18000	2.86 is the gain percent.	2.86 is the gain percent.	✔

Ex. No. : 1.5 **Date: 14/02/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Deposits

In many jurisdictions, a small deposit is added to drink containers to encourage people to recycle them. In one particular jurisdiction, drink containers holding one liter or less have a $0.10 deposit and drink containers holding more than one liter have a $0.25 deposit. Write a program that reads the number of containers of each size(less and more) from the user. Your program should continue by computing and displaying the refund that will be received for returning those containers. Format the output so that it includes a dollar sign and always displays exactly two decimal places.

Sample Input

10

20

Sample Output

Your total refund will be $6.00.

For example:

Input	Result
20 20	Your total refund will be $7.00.

Program:

```
less=int(input())
more=int(input())
tot=less*0.10+more*0.25
print("Your total refund will be $",format(tot,'.2f'),sep='',end='.')
```

	Input	Expected	Got	
✔	20 20	Your total refund will be $7.00.	Your total refund will be $7.00.	✔
✔	11 22	Your total refund will be $6.60.	Your total refund will be $6.60.	✔
✔	123 200	Your total refund will be $62.30.	Your total refund will be $62.30.	✔
✔	76 38	Your total refund will be $17.10.	Your total refund will be $17.10.	✔

Ex. No. : 1.6 **Date: 14/02/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Carpenter

Justin is a carpenter who works on an hourly basis. He works in a company where he is paid Rs 50 for an hour on weekdays and Rs 80 for an hour on weekends. He works 10 hrs more on weekdays than weekends. If the salary paid for him is given, write a program to find the number of hours he has worked on weekdays and weekends.

Hint:

If the final result(hrs) are in -ve convert that to +ve using abs() function

The abs() function returns the absolute value of the given number.

```
number = -20
absolute_number = abs(number)
print(absolute_number)
# Output: 20
```

Sample Input:

450

Sample Output:

weekdays 10.38

weekend 0.38

For example:

Input	Result
450	weekdays 10.38 weekend 0.38

Program:

```
sal=int(input())
x=(sal-500)/130
absolute=abs(x)
print("weekdays",format(absolute+10,'.2f'))
print("weekend",format(absolute,'.2f')
```

	Input	Expected	Got	
✔	450	weekdays 10.38 weekend 0.38	weekdays 10.38 weekend 0.38	✔
✔	500	weekdays 10.00 weekend 0.00	weekdays 10.00 weekend 0.00	✔
✔	10000	weekdays 83.08 weekend 73.08	weekdays 83.08 weekend 73.08	✔
✔	6789	weekdays 58.38 weekend 48.38	weekdays 58.38 weekend 48.38	✔

02- Operators in Python

Ex. No. : 2.1 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Widgets and Gizmos

An online retailer sells two products: widgets and gizmos. Each widget weighs 75 grams. Each gizmo weighs 112 grams. Write a program that reads the number of widgets and the number of gizmos from the user. Then your program should compute and display the total weight of the parts.

Sample Input

10

20

Sample Output

The total weight of all these widgets and gizmos is 2990 grams.

For example:

Input	Result
10 20	The total weight of all these widgets and gizmos is 2990 grams.

Program:

```
w=int(input())
g=int(input())
print("The total weight of all these widgets and gizmos is",(w*75)+(g*112),"grams.")
```

	Input	Expected
✔	10 20	The total weight of all these widgets and gizmos is 2990 grams.

Ex. No. : 2.2 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Doll Sings

In London, every year during Dasara there will be a very grand doll show. People try to invent new dolls of different varieties. The best-sold doll's creator will be awarded with a cash prize. So people broke their heads to create dolls innovatively. Knowing this competition, Mr.Lokpaul tried to create a doll that sings only when an even number is pressed and the number should not be zero and greater than 100.

IF Lokpaul wins print true, otherwise false.

Sample Input

10

Sample Output

True

Explanation:

Since 10 is an even number and a number between 0 and 100, True is printed

Program:

```
n=int(input())
if(n%2==0 and n!=0 and n<100):
    print("True")
else:
    print("False")
```

	Input	Expected	Got	
✔	56	True	True	✔
✔	101	False	False	✔
✔	-1	False	False	✔

Ex. No. : 2.3 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Birthday Party

Mr. X's birthday is in next month. This time he is planning to invite N of his friends. He wants to distribute some chocolates to all of his friends after the party. He went to a shop to buy a packet of chocolates. At the chocolate shop, 4 packets are there with different numbers of chocolates. He wants to buy such a packet which contains a number of chocolates, which can be distributed equally among all of his friends. Help Mr. X to buy such a packet.

Input Given:

N-No of friends

P1,P2,P3 AND P4-No of chocolates

OUTPUT:

"True" if he can buy that packet and "False" if he can't buy that packet.

SAMPLE INPUT AND OUTPUT:

5

25

12

10

9

OUTPUT

True False True False

Program:

```
n=int(input())
p1=int(input())
p2=int(input())
p3=int(input())
p4=int(input())
I=(p1,p2,p3,p4)
for i in I:
```

```
if(i%n==0):
    print("True",end=" ")
else:
    print("False",end=" ")
```

	Input	Expected	Got	
✔	5 25 23 20 10	True False True True	True False True True	✔
✔	4 23 24 21 12	False True False True	False True False True	✔
✔	8 64 8 16 32	True True True True	True True True True	✔

Ex. No. : 2.4 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Hamming Weight

Write a python program that takes a integer between 0 and 15 as input and displays the number of '1' s in its binary form.(Hint:use python bitwise operator.

Sample Input

3

Sample Output:

2

Explanation:

The binary representation of 3 is 011, hence there are 2 ones in it. so the output is 2.

Program:

```
def count_ones(n):
    c=0
    while(n):
        c+=n&1
        n>>=1
    return c
num=int(input())
ones_count=count_ones(num)
print(ones_count)
```

	Input	Expected	Got	
✔	3	2	2	✔
✔	5	2	2	✔
✔	15	4	4	✔

Ex. No. : 2.5 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Compound Interest

Pretend that you have just opened a new savings account that earns 4 percent interest per year. The interest that you earn is paid at the end of the year, and is added to the balance of the savings account. Write a program that begins by reading the amount of money deposited into the account from the user. Then your program should compute and display the amount in the savings account after 1, 2, and 3 years. Display each amount so that it is rounded to 2 decimal places.

Sample Input:

10000

Sample Output:

Balance as of end of Year 1: $10400.00.

Balance as of end of Year 2: $10816.00.

Balance as of end of Year 3: $11248.64

Program:

```
a=int(input())
y1=a+(a*0.04)
y2=y1+(y1*0.04)
y3=y2+(y2*0.04)
print("Balance as of end of Year 1: ",end='$')
print(format(y1,'.2f'),end=".")
print("\nBalance as of end of Year 2: ",end='$')
print(format(y2,'.2f'),end=".")
print("\nBalance as of end of Year 3: ",end='$')
print(format(y3,'.2f'),end=".")
```

	Input	Expected	Got	
✔	10000	Balance as of end of Year 1: $10400.00. Balance as of end of Year 2: $10816.00. Balance as of end of Year 3: $11248.64.	Balance as of end of Year 1: $10400.00. Balance as of end of Year 2: $10816.00. Balance as of end of Year 3: $11248.64.	✔
✔	20000	Balance as of end of Year 1: $20800.00. Balance as of end of Year 2: $21632.00. Balance as of end of Year 3: $22497.28.	Balance as of end of Year 1: $20800.00. Balance as of end of Year 2: $21632.00. Balance as of end of Year 3: $22497.28.	✔

Ex. No. : 2.6 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Eligible to donate blood

A team from the Rotract club had planned to conduct a rally to create awareness among the Coimbatore people to donate blood. They conducted the rally successfully. Many of the Coimbatore people realized it and came forward to donate their blood to nearby blood banks. The eligibility criteria for donating blood are people should be above or equal to 18 and his/ her weight should be above 40. There was a huge crowd and staff in the blood bank found it difficult to manage the crowd. So they decided to keep a system and ask the people to enter their age and weight in the system. If a person is eligible he/she will be allowed inside.

Write a program and feed it to the system to find whether a person is eligible or not.

Input Format:

Input consists of two integers that correspond to the age and weight of a person respectively.

Output Format:

Display True(IF ELIGIBLE)

Display False (if not eligible)

Sample Input

19

45

Sample Output

True

Program:

```
a=int(input())

w=int(input())

eligible=(a>=18)&(w>40)

print(eligible)
```

	Input	Expected	Got	
✔	19 45	True	True	✔
✔	18 40	False	False	✔
✔	18 42	True	True	✔
✔	16 45	False	False	✔

Ex. No. : 2.7 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

C or D

Mr.Ram has been given a problem kindly help him to solve it. The input of the program is either 0 or 1. IF 0 is the input he should display "C" if 1 is the input it should display "D".There is a constraint that Mr. Ram should use either logical operators or arithmetic operators to solve the problem, not anything else.

Hint:

Use ASCII values of C and D.

Input Format:

An integer x, 0<=x<=1. .

Output Format:

output a single character "C" or "D"depending on the value of x.

Input 1:
0
Output 1:

C

Input 2:
1

Output 1:
D

Program:

```
a=int(input())
c=chr(a+67)
print(c)
```

	Input	Expected	Got	
✔	0	C	C	✔
✔	1	D	D	✔

Ex. No. : 2.8 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Troy Battle

In the 1800s, the battle of Troy was led by Hercules. He was a superstitious person. He believed that his crew can win the battle only if the total count of the weapons in hand is in multiple of 3 and the soldiers are in an even number of count. Given the total number of weapons and the soldier's count, Find whether the battle can be won or not according to Hercules's belief. If the battle can be won print True otherwise print False.

Input format:

Line 1 has the total number of weapons

Line 2 has the total number of Soldiers.

Output Format:

If the battle can be won print True otherwise print False.

Sample Input:

32

43

Sample Output:'

False

Program:

```
w=int(input())

s=int(input())

if(w%3==0 and s%2==0):

    print("True")

else:
```

print("False")

	Input	Expected	Got	
✔	32 43	False	False	✔
✔	273 7890	True	True	✔
✔	800 4590	False	False	✔
✔	6789 32996	True	True	✔

Ex. No. : 2.9 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Tax and Tip

The program that you create for this exercise will begin by reading the cost of a meal ordered at a restaurant from the user. Then your program will compute the tax and tip for the meal. Use your local tax rate (5 percent) when computing the amount of tax owing. Compute the tip as 18 percent of the meal amount (without the tax). The output from your program should include the tax amount, the tip amount, and the grand total for the meal including both the tax and the tip. Format the output so that all of the values are displayed using two decimal places.

Sample Input

100

Sample Output

The tax is 5.00 and the tip is 18.00, making the total 123.00

Program:

```
a=int(input())
tax=0.05
tip=0.18
c=tax*a
d=tip*a
print("The tax is",format(c,'.2f'),"and the tip is",format(d,'.2f'),end=",")
print(" making the total",format((tax+tip)*a+a,'.2f'))
```

	Input	Expected
✔	100	The tax is 5.00 and the tip is 18.00, making the total 123.00
✔	250	The tax is 12.50 and the tip is 45.00, making the total 307.50

Ex. No. : 2.10 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Return last digit of the given number

Write a program that returns the last digit of the given number. Last digit is being referred to the least significant digit i.e. the digit in the ones (units) place in the given number.

The last digit should be returned as a positive number.

For example,

if the given number is 197, the last digit is 7

if the given number is -197, the last digit is 7

For example:

Input	Result
123	3

Program:

```
a=int(input())
if(a<0):
    b=a*(-1)
    c=b%10
    print(c)
else:
    c=a%10
    print(c)
```

	Input	Expected	Got	
✔	197	7	7	✔
✔	-197	7	7	✔

03 - Selection Structures in Python

Ex. No. : 3.1 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Admission Eligibility

Write a program to find the eligibility of admission for a professional course based on the following criteria:

Marks in Maths >= 65

Marks in Physics >= 55

Marks in Chemistry >= 50

Or

Total in all three subjects >= 180

Sample Test Cases

Test Case 1

Input

70

60

80

Output

The candidate is eligible

Test Case 2

Input

50

80

80

Output

The candidate is eligible

Test Case 3

Input

50

60

40

Output

The candidate is not eligible

For example:

Input	Result
50 80 80	The candidate is eligible

Program:

```
mat=int(input())
phy=int(input())
che=int(input())
tot=mat+phy+che
if mat>=65 and phy>=55 and che>=50:
    print("The candidate is eligible")
elif tot>=180:
    print("The candidate is eligible")
else:
    print("The candidate is not eligible")
```

	Input	Expected	Got	
✔	70 60 80	The candidate is eligible	The candidate is eligible	✔
✔	50 80 80	The candidate is eligible	The candidate is eligible	✔
✔	50 60 40	The candidate is not eligible	The candidate is not eligible	✔
✔	20 10 25	The candidate is not eligible	The candidate is not eligible	✔

Ex. No. : 3.2 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Classifying Triangles

A triangle can be classified based on the lengths of its sides as equilateral, isosceles or scalene. All three sides of an equilateral triangle have the same length. An isosceles triangle has two sides that are the same length, and a third side that is a different length. If all of the sides have different lengths then the triangle is scalene.

Write a program that reads the lengths of the three sides of a triangle from the user. Then display a message that states the triangle's type.

Sample Input 1

60

60

60

Sample Output 1

That's a equilateral triangle

For example:

Input	Result
40 40 80	That's a isosceles triangle

Program:

```
a=int(input())

b=int(input())

c=int(input())

if a==b==c:

    print("That's a equilateral triangle")
```

```
elif a==b or a==c or b==c:
    print("That's a isosceles triangle")
else:
    print("That's a scalene triangle")
```

	Input	Expected	Got	
✔	60 60 60	That's a equilateral triangle	That's a equilateral triangle	✔
✔	40 40 80	That's a isosceles triangle	That's a isosceles triangle	✔
✔	50 60 70	That's a scalene triangle	That's a scalene triangle	✔
✔	50 50 80	That's a isosceles triangle	That's a isosceles triangle	✔
✔	10 10 10	That's a equilateral triangle	That's a equilateral triangle	✔

Ex. No. : 3.3 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Electricity Bill

Write a program to calculate and print the Electricity bill where the unit consumed by the user is given from test case. It prints the total amount the customer has to pay. The charge are as follows:

Unit	Charge / Unit
Upto 199	@1.20
200 and above but less than 400	@1.50
400 and above but less than 600	@1.80
600 and above	@2.00

If bill exceeds Rs.400 then a surcharge of 15% will be charged and the minimum bill should be of Rs.100/-

Sample Test Cases

Test Case 1

Input

50

Output

100.00

Test Case 2

Input

300

Output

517.50

For example:

Input	Result

Input	Result
500	1035.00

Program:

```
u=float(input())
if u<=199:
    r=u*1.20
    if(r<100):
        r=100
elif u>=200 and u<400:
    r=u*1.50
elif u>=400 and u<600:
    r=u*1.80
elif u>=600:
    r=u*2.00
if r>400:
    r=r+(r*15/100)
print(format(r,'.2f'))
```

	Input	Expected	Got	
✔	50	100.00	100.00	✔
✔	100.00	120.00	120.00	✔
✔	500	1035.00	1035.00	✔
✔	700	1610.00	1610.00	✔

Passed all tests! ✔

Ex. No. : 3.4 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

IN/OUT

Ms. Sita, the faculty handling programming lab for you is very strict. Your seniors have told you that she will not allow you to enter the week's lab if you have not completed atleast half the number of problems given last week. Many of you didn't understand this statement and so they requested the good programmers from your batch to write a program to find whether a student will be allowed into a week's lab given the number of problems given last week and the number of problems solved by the student in that week.

Input Format:

Input consists of 2 integers.

The first integer corresponds to the number of problems given and the second integer corresponds to the number of problems solved.

Output Format:

Output consists of the string “IN” or “OUT”.

Sample Input and Output:

Input

8

3

Output

OUT

For example:

Input	Result
8 3	OUT

Program:

```
a=int(input())
b=int(input())
if b>=a/2:
    print("IN")
else:
    print("OUT")
```

	Input	Expected	Got	
✔	8 3	OUT	OUT	✔
✔	8 5	IN	IN	✔
✔	20 9	OUT	OUT	✔
✔	50 31	IN	IN	✔

Passed all tests! ✔

Ex. No. : 3.5 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Vowel or Consonant

In this exercise you will create a program that reads a letter of the alphabet from the user. If the user enters a, e, i, o or u then your program should display a message indicating that the entered letter is a vowel. If the user enters 'y' then your program should display a message indicating that sometimes y is a vowel, and sometimes y is a consonant. Otherwise your program should display a message indicating that the letter is a consonant.

Sample Input 1

i

Sample Output 1

It's a vowel.

Sample Input 2

y

Sample Output 2

Sometimes it's a vowel... Sometimes it's a consonant.

Sample Input3

c

Sample Output 3

It's a consonant.

For example:

Input	Result
y	Sometimes it's a vowel... Sometimes it's a consonant.
u	It's a vowel.
p	It's a consonant.

Program:

```
char=input()
if char=='y':
    print("Sometimes it's a vowel... Sometimes it's a consonant.")
elif char=='a' or char=='e' or char=='i' or char=='o' or char=='u':
    print("It's a vowel.")
else:
    print("It's a consonant.")
```

	Input	Expected
✔	i	It's a vowel.
✔	y	Sometimes it's a vowel... Sometimes it's a consonant.
✔	c	It's a consonant.
✔	e	It's a vowel.
✔	r	It's a consonant.

Ex. No. : 3.6 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Leap Year

Most years have 365 days. However, the time required for the Earth to orbit the Sun is actually slightly more than that. As a result, an extra day, February 29, is included in some years to correct for this difference. Such years are referred to as leap years. The rules for determining whether or not a year is a leap year follow:

- Any year that is divisible by 400 is a leap year.
- Of the remaining years, any year that is divisible by 100 is not a leap year.
- Of the remaining years, any year that is divisible by 4 is a leap year.
- All other years are not leap years.

Write a program that reads a year from the user and displays a message indicating whether or not it is a leap year.

Sample Input 1

1900

Sample Output 1

1900 is not a leap year.

Sample Input 2

2000

Sample Output 2

2000 is a leap year.

Program:

```
year=int(input())
if  year%4==0 and year%400==0:
    print(year,"is a leap year.")
```

```
elif year%100==0:
    print(year,"is not a leap year.")
else:
    print(year,"is not a leap year.")
```

	Input	Expected	Got	
✔	1900	1900 is not a leap year.	1900 is not a leap year.	✔
✔	2000	2000 is a leap year.	2000 is a leap year.	✔
✔	2100	2100 is not a leap year.	2100 is not a leap year.	✔
✔	2400	2400 is a leap year.	2400 is a leap year.	✔

Ex. No. : 3.7 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Month name to days

The length of a month varies from 28 to 31 days. In this exercise you will create a program that reads the name of a month from the user as a string. Then your program should display the number of days in that month. Display “28 or 29 days” for February so that leap years are addressed.

Sample Input 1

February

Sample Output 1

February has 28 or 29 days in it.

Sample Input 2

March

Sample Output 2

March has 31 days in it.

Sample Input 3

April

Sample Output 3

April has 30 days in it.

For example:

Input	Result
February	February has 28 or 29 days in it.
March	March has 31 days in it.

Program:

```
m=input()
d=0
if m=="January":
    d=31
elif m=="March":
    d=31
elif m=="April":
    d=30
elif m=="May":
    d=31
elif m=="June":
    d=30
elif m=="July":
    d=31
elif m=="August":
    d=31
elif m=="September":
    d=30
elif m=="October":
    d=31
elif m=="November":
    d=30
elif m=="December":
```

```
    d=31
if(d!=0):
    print(m,"has",d,"days in it.")
elif m=="February":
    print("February has 28 or 29 days in it.")
```

	Input	Expected	Got	
✔	February	February has 28 or 29 days in it.	February has 28 or 29 days in it.	✔
✔	March	March has 31 days in it.	March has 31 days in it.	✔
✔	April	April has 30 days in it.	April has 30 days in it.	✔
✔	May	May has 31 days in it.	May has 31 days in it.	✔

Passed all tests! ✔

Ex. No. : 3.8 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Pythagorean triple

Three numbers form a Pythagorean triple if the sum of squares of two numbers is equal to the square of the third.

For example, 3, 5 and 4 form a Pythagorean triple, since 3*3 + 4*4 = 25 = 5*5 You are given three integers, a, b, and c. They need not be given in increasing order. If they form a Pythagorean triple, then print "Yes", otherwise, print "No".

Sample Input

3

5

4

Sample Output

Yes

For example:

Input	Result
3 4 5	Yes

Program:

```
a=int(input())

b=int(input())

c=int(input())

if a**2+b**2==c**2 or b**2+c**2==a**2 or a**2+c**2==b**2:

    print("yes")
```

else:

print("no")

	Input	Expected	Got	
✔	3 5 4	yes	yes	✔
✔	5 8 2	no	no	✔

Passed all tests! ✔

Ex. No. : 3.9 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Second last digit

Write a program that returns the second last digit of the given number. Second last digit is being referred 10the digit in the tens place in the given number.

For example, if the given number is 197, the second last digit is 9.

Note1 - The second last digit should be returned as a positive number. i.e. if the given number is -197, the second last digit is 9.

Note2 - If the given number is a single digit number, then the second last digit does not exist. In such cases, the program should return -1. i.e. if the given number is 5, the second last digit should be returned as -1.

For example:

Input	Result
197	9

Program:

```
x=int(input())
if x<0:
  x=abs(x)
if x<10:
   print("-1")
else:
   print(x%100//10)
```

	Input	Expected	Got	
✔	197	9	9	✔
✔	-197	9	9	✔
✔	5	-1	-1	✔
✔	123456	5	5	✔
✔	8	-1	-1	✔

Passed all tests! ✔

Ex. No. : 3.10 **Date: 4/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Chinese Zodiac

The Chinese zodiac assigns animals to years in a 12 year cycle. One 12 year cycle is shown in the table below. The pattern repeats from there, with 2012 being another year of the dragon, and 1999 being another year of the hare.

Year Animal

2000 Dragon

2001 Snake

2002 Horse

2003 Sheep

2004 Monkey

2005 Rooster

2006 Dog

2007 Pig

2008 Rat

2009 Ox

2010 Tiger

2011 Hare

Write a program that reads a year from the user and displays the animal associated with that year. Your program should work correctly for any year greater than or equal to zero, not just the ones listed in the table.

Sample Input 1

2010

Sample Output 1

2010 is the year of the Tiger.

Sample Input 2

2020

Sample Output 2

2020 is the year of the Rat.

Program:

```
y=int(input())
x=(y%2000)%12
if x==0:
    c="Dragon"
elif x==1:
    c="Snake"
elif x==2:
    c="Horse"
elif x==3:
    c="Sheep"
elif x==4:
    c="Monkey"
elif x==5:
    c="Rooster"
elif x==6:
    c="Dog"
elif x==7:
    c="Pig"
elif x==8:
    c="Rat"
elif x==9:
```

```
    c="Ox"
elif x==10:
    c="Tiger"
elif x==11:
    c="Hare"
print(y,"is the year of the",c,end='.')
```

	Input	Expected	Got	
✔	2010	2010 is the year of the Tiger.	2010 is the year of the Tiger.	✔
✔	2020	2020 is the year of the Rat.	2020 is the year of the Rat.	✔

Passed all tests! ✔

04 - Iteration Control Structures

Ex. No. : 4.1 **Date: 14/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Factors of a number

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number).

For example:

Input	Result
20	1 2 4 5 10 20

Program:

```
a=int(input())

for i in range(1,a+1):

    if a%i==0:

        print(i, end=")
```

	Input	Expected	Got	
✔	20	1 2 4 5 10 20	1 2 4 5 10 20	✔
✔	5	1 5	1 5	✔
✔	13	1 13	1 13	✔

Passed all tests! ✔

Ex. No. : 4.2 **Date: 14/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Non Repeated Digit Count

Write a program to find the count of non-repeated digits in a given number N. The number will be passed to the program as an input of type int.
Assumption: The input number will be a positive integer number >= 1 and <= 25000.
Some examples are as below.
If the given number is 292, the program should return 1 because there is only 1 non--repeated digit '9' in this number
If the given number is 1015, the program should return 2 because there are 2 non-repeated digits in this number, '0', and '5'.
If the given number is 108, the program should return 3 because there are 3 non--repeated digits in this number, '1', '0', and '8'.
If the given number is 22, the function should return 0 because there are NO non--repeated digits in this number.

For example:

Input	Result
292	1
1015	2
108	3
22	0

Program:

```
n=int(input())

digits=str(n)

repeat=set()

unique=set()
```

```
for i in digits:
    if i in unique:
        repeat.add(i)
    else:
        unique.add(i)
count=len(unique-repeat)
print(count)
```

	Input	Expected	Got	
✔	292	1	1	✔
✔	1015	2	2	✔
✔	108	3	3	✔
✔	22	0	0	✔

Passed all tests! ✔

Ex. No. : 4.3 **Date: 14/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Prime Checking

Write a program that finds whether the given number N is Prime or not. If the number is prime, the program should return 2 else it must return 1.

Assumption: 2 <= N <=5000, where N is the given number.

Example1: if the given number N is 7, the method must return 2

Example2: if the given number N is 10, the method must return 1

For example:

Input	Result
7	2
10	1

Program:

```
N=int(input())

if N==1:

    print("1")

elif N>1:
  for i in range(2,n):
      if(N%i)==0:
            print("1")

            break

       else:
          print("2")
```

	Input	Expected	Got	
✔	7	2	2	✔
✔	10	1	1	✔

Passed all tests! ✔

Ex. No. : 4.4 **Date: 14/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Next Perfect Square

Given a number N, find the next perfect square greater than N.

Input Format:

Integer input from stdin.

Output Format:

Perfect square greater than N.

Example Input:

10

Output:

16

Program:

```
from math import sqrt
 n=int(input())
while int(sqrt(n))!=sqrt(n):
  n=n+1
  print(n)
```

	Input	Expected	Got	
✔	10	16	16	✔

Passed all tests! ✔

Ex. No. : 4.5 **Date: 14/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Nth Fibonacci

Write a program to return the nth number in the fibonacci series. The value of N will be passed to the program as input.

NOTE: Fibonacci series looks like –

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . . and so on.

i.e. Fibonacci series starts with 0 and 1, and continues generating the next number as the sum of the previous two numbers.

- first Fibonacci number is 0,
- second Fibonacci number is 1,
- third Fibonacci number is 1,
- fourth Fibonacci number is 2,
- fifth Fibonacci number is 3,
- sixth Fibonacci number is 5,
- seventh Fibonacci number is 8, and so on.

For example:

Input:

7

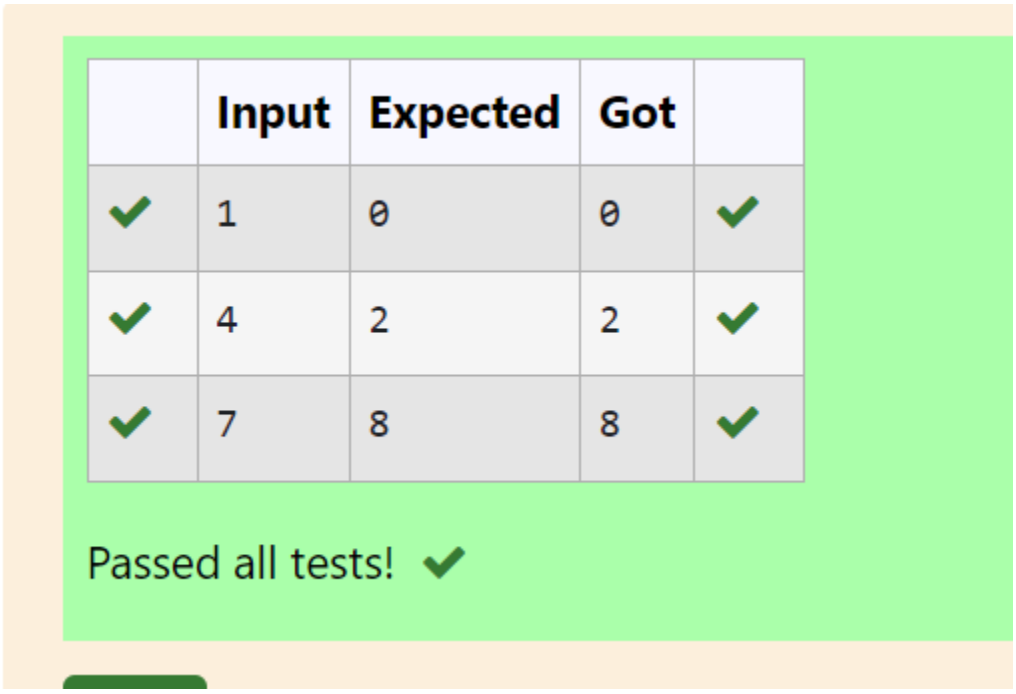
Output

8

Program:

```
n=int(input())
if n<2:

  print(n-1)

else:
   n=n-1
   fs=[0,1]
```

```
for i in range(1,n):
    fs.append(fs[i]+fs [i-1])
print(fs [n])
```



	Input	Expected	Got	
✔	1	0	0	✔
✔	4	2	2	✔
✔	7	8	8	✔

Passed all tests! ✔

Ex. No. : 4.6 **Date: 14/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Disarium Number

A Number is said to be Disarium number when the sum of its digit raised to the power of their respective positions becomes equal to the number itself. Write a program to print number is Disarium or not.

Input Format:

Single Integer Input from stdin.

Output Format:

Yes or No.

Example Input:

175

Output:

Yes

Explanation

1^1 + 7^2 +5^3 = 175

Example Input:

123

Output:

No

For example:

Input	Result
175	Yes
123	No

Program:

```
n=int(input())

digits=str(n)
```

```
sum=0
power-1
for i in digits:
        sum+=int(i)**power
        power+=1
if sum==n:
  print("Yes")
else:
  print("No")
```

	Input	Expected	Got	
✔	175	Yes	Yes	✔
✔	123	No	No	✔

Passed all tests! ✔

Ex. No. : 4.7 **Date: 14/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Sum of Series

Write a program to find the sum of the series 1 +11 + 111 + 1111 + . . . + n terms (n will be given as input from the user and sum will be the output)

Sample Test Cases

Test Case 1

Input

4

Output

1234

Explanation:

as input is 4, have to take 4 terms.

1 + 11 + 111 + 1111

Test Case 2

Input

6

Output

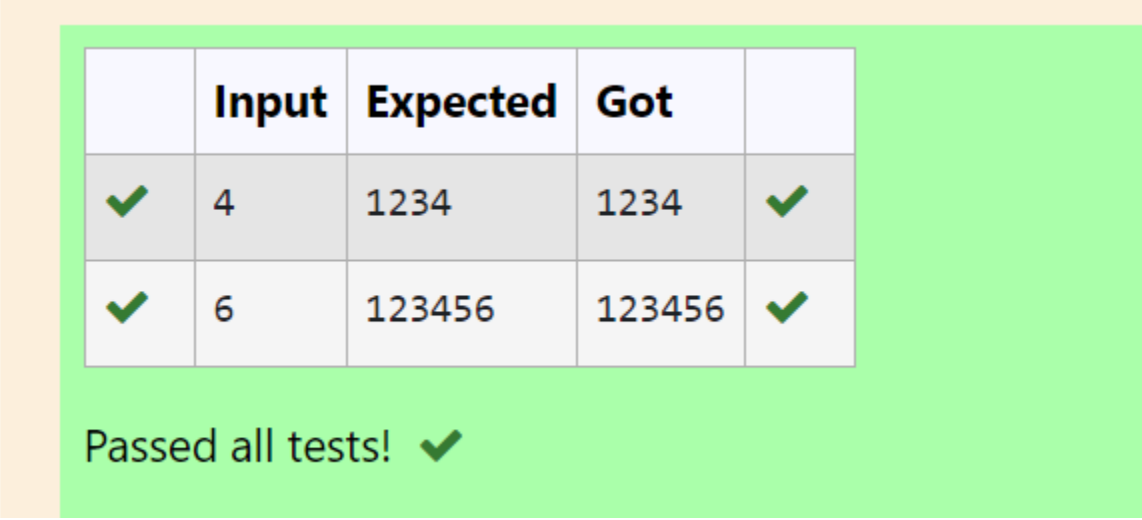
123456

For example:

Input	Result
3	123

Program:

```
n=int(input())
sum=0
term=1
for i in range(1,n+1):
    sum+-term
    term-term*10+1
print(sum)
```



	Input	Expected	Got	
✔	4	1234	1234	✔
✔	6	123456	123456	✔

Passed all tests! ✔

Ex. No. : 4.8 **Date: 14/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Unique Digit Count

Write a program to find the count of unique digits in a given number N. The number will be passed to the program as an input of type int.
Assumption: The input number will be a positive integer number >= 1 and <= 25000.
For e.g.
If the given number is 292, the program should return 2 because there are only 2 unique digits '2' and '9' in this number
If the given number is 1015, the program should return 3 because there are 3 unique digits in this number, '1', '0', and '5'.
For example:

Input	Result
292	2
1015	3

Program:

```
def count(n):
  digits=str(n)
  unique=list()
  for i in digits:
     if i not in unique:
          unique.append(i)
   return len(unique)
n=int(input())
print(count(n))
```

	Input	Expected	Got	
✔	292	2	2	✔
✔	1015	3	3	✔
✔	123	3	3	✔

Passed all tests! ✔

Ex. No. : 4.9 **Date: 14/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Product of single digit

Given a positive integer N, check whether it can be represented as a product of single digit numbers.

Input Format:

Single Integer input.

Output Format:

Output displays Yes if condition satisfies else prints No.

Example Input:

14

Output:

Yes

Example Input:

13

Output:

No

Program:

```
a=int(input())
if a%2==0 or a%3==0 or a%5==0 or a%7==0 or a%9==0:
  print("Yes")
else:
  print("No")
```

	Input	Expected	Got	
✔	14	Yes	Yes	✔
✔	13	No	No	✔

Passed all tests! ✔

Correct

Ex. No. : 4.10 **Date: 14/04/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Perfect Square After adding One

Given an integer N, check whether N the given number can be made a perfect square after adding 1 to it.

Input Format:

Single integer input.

Output Format:

Yes or No.

Example Input:

24

Output:

Yes

Example Input:

26

Output:

No

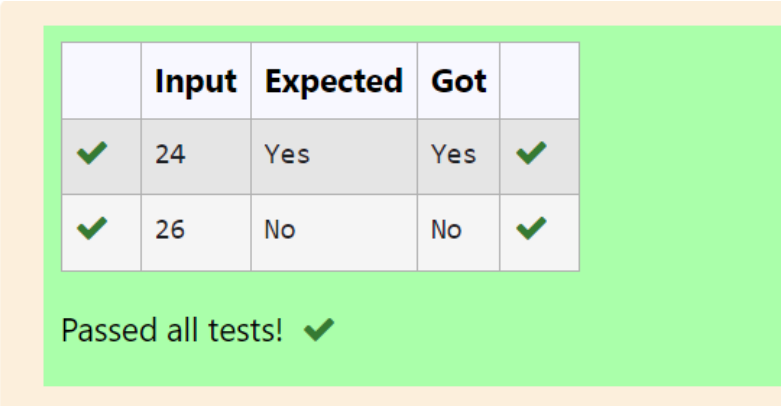
For example:

Input	Result
24	Yes

Program:

```
import math

a=int(input())

b=math.sqrt(a+1)
```

```
if b.is_integer():
    print("Yes")
else:
    print("No")
```



	Input	Expected	Got	
✔	24	Yes	Yes	✔
✔	26	No	No	✔

Passed all tests! ✔

05 - Strings in Python

Ex. No. : 5.1 **Date: 2/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Count Chars

Write a python program to count all letters, digits, and special symbols respectively from a given string

For example:

Input Result
rec@123
3
3
1

```
l=0
d=0
s=0
str=input()
for i in str:
    if i.isalpha():
        l+=1
    elif i.isdigit():
        d+=1
    else:
        s+=1
print(l)
print(d)
print(s)
```

	Input	Expected	Got	
✔	rec@123	3 3 1	3 3 1	✔
✔	P@#yn26at^&i5ve	8 3 4	8 3 4	✔
✔	abc@12&	3 2 2	3 2 2	✔

Passed all tests! ✔

Ex. No. : 5.2 **Date: 2/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Decompress the String

Assume that the given string has enough memory. Don't use any extra space(IN-PLACE)

Sample Input 1
a2b4c6

Sample Output 1
aabbbbccccccc

```
input_string=input()
result = ""
i = 0
while i < len(input_string):
    char = input_string[i]
    if i + 1 < len(input_string) and input_string[i + 1].isdigit():
        j = i + 1
        while j < len(input_string) and input_string[j].isdigit():
            j += 1
        count = int(input_string[i + 1:j])
        result += char * count
        i = j
    else:
        result += char
        i += 1
print(result)
```

	Input	Expected	Got	
✔	a2b4c6	aabbbbccccccc	aabbbbccccccc	✔
✔	a12b3d4	aaaaaaaaaaaabbbdddd	aaaaaaaaaaaabbbdddd	✔

Passed all tests! ✔

Ex. No. : 5.3 **Date: 2/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

First N Common Chars

Two string values S1, S2 are passed as the input. The program must print first N characters present in S1 which are also present in S2.

Input Format:

The first line contains S1.
The second line contains S2.
The third line contains N.

Output Format:

The first line contains the N characters present in S1 which are also present in S2.

Boundary Conditions:

2 <= N <= 10
2 <= Length of S1, S2 <= 1000

Example Input/Output 1:

Input:

abcbde
cdefghbb
3

Output:

bcd

Note:

b occurs twice in common but must be printed only once.

s1=input()

s2=input()

```
n=int(input())
common_chars=set(s1)&set(s2)
result=""
for char in s1:
    if char in common_chars:
        result+=char
        common_chars.remove(char)
    if len(result)==n:
        break
print(result[:n])
```

	Input	Expected	Got	
✔	abcbde cdefghbb 3	bcd	bcd	✔

Passed all tests! ✔

Ex. No. : 5.4 **Date: 2/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Remove Characters

Given two Strings s1 and s2, remove all the characters from s1 which is present in s2.

Constraints
1<= string length <= 200

Sample Input 1
experience
enc

Sample Output 1
xpri

```
s1=input()
s2=input()
s2_set = set(s2)
result = ""
for char in s1:
    if char not in s2_set:
        result += char
print(result)
```

	Input	Expected	Got	
✔	experience enc	xpri	xpri	✔

Passed all tests! ✔

Ex. No. : 5.5 **Date: 2/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Remove Palindrome Words

String should contain only the words are not palindrome.

Sample Input 1
Malayalam is my mother tongue

Sample Output 1
is my mother tongue

```
def is_palindrome (word):
    return word == word[::-1]
def filter_non_palindromic_words(input_string):
    words = input_string.split()
    non_palindromic_words = [word for word in words if not is_palindrome (word)]
    return ' '.join(non_palindromic_words)
input_string = input().lower()
output_string = filter_non_palindromic_words (input_string)
print(output_string)
```

	Input	Expected	Got	
✔	Malayalam is my mother tongue	is my mother tongue	is my mother tongue	✔

Passed all tests! ✔

Ex. No. : 5.6 **Date: 2/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Return Second World in Uppercase

Write a program that takes as input a string (sentence), and returns its second word in uppercase.

For example:

If input is "Wipro Technologies Bangalore" the function should return "TECHNOLOGIES"
If input is "Hello World" the function should return "WORLD"
If input is "Hello" the program should return "LESS"

NOTE 1: If input is a sentence with less than 2 words, the program should return the word "LESS".
NOTE 2: The result should have no leading or trailing spaces.

For example:

Input Result
Wipro Technologies Bangalore
TECHNOLOGIES
Hello World
WORLD
Hello
LESS

```
sentence=input()
words=sentence.split()
if len(words)<2:
  print("LESS")
else:
  print(words[1].upper())
```

	Input	Expected	Got	
✔	Wipro Technologies Bangalore	TECHNOLOGIES	TECHNOLOGIES	✔
✔	Hello World	WORLD	WORLD	✔
✔	Hello	LESS	LESS	✔

Passed all tests! ✔

Ex. No. : 5.7 **Date: 2/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Revers String

Reverse a string without affecting special characters. Given a string S, containing special characters and all the alphabets, reverse the string without affecting the positions of the special characters.

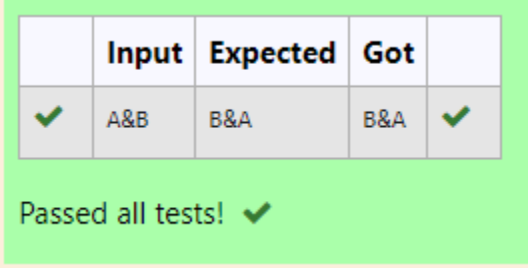
Input:
A&B
Output:
B&A
Explanation: As we ignore '&' and
As we ignore '&' and then reverse, so answer is "B&A".

For example:

Input Result
A&x#
x&A#

```
string=input()
string_list = list(string)
left = 0
right = len(string_list) - 1
while left < right:
    if not string_list[left].isalpha():
        left += 1
        continue
    if not string_list[right].isalpha():
        right -= 1
        continue
    string_list[left], string_list[right] = string_list[right], string_list[left]
```

```
        left += 1
        right -= 1
reversed_string = ''.join(string_list)
print(reversed_string)
```



	Input	Expected	Got	
✔	A&B	B&A	B&A	✔

Passed all tests! ✔

Ex. No. : 5.8 **Date: 2/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

String characters balance Test

Write a program to check if two strings are balanced. For example, strings s1 and s2 are balanced if all the characters in the s1 are present in s2. The character's position doesn't matter. If balanced display as "true" ,otherwise "false".

For example:

Input Result
Yn
PYnative
True

```
s1=input()
s2=input()
s1=s1.lower()
s2=s2.lower()
def balance(s1,s2):
    for i in s1:
        if i not in s2:
            return False
    return True
print(balance(s1,s2))
```

	Input	Expected	Got	
✔	Yn PYnative	True	True	✔
✔	Ynf PYnative	False	False	✔

Passed all tests! ✔

Ex. No. : 5.9 **Date: 2/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Unique Names

In this exercise, you will create a program that reads words from the user until the user enters a blank line. After the user enters a blank line your program should display each word entered by the user exactly once. The words should be displayed in the same order that they were first entered. For example, if the user enters:

Input:

first
second
first
third
second

then your program should display:

Output:
first
second
third

```
a=set()
for i in iter(input, ""):
    if i not in a:
        a.add(i)
        print(i)
```

	Input	Expected	Got	
✔	first second first third second	first second third	first second third	✔
✔	rec cse it rec cse	rec cse it	rec cse it	✔

Passed all tests! ✔

Ex. No. : 5.10 **Date: 2/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Username Domain Extension

Given a string S which is of the format USERNAME@DOMAIN.EXTENSION, the program must print the EXTENSION, DOMAIN, USERNAME in the reverse order.

Input Format:
The first line contains S.

Output Format:
The first line contains EXTENSION.
The second line contains DOMAIN.
The third line contains USERNAME.

Boundary Condition:
1 <= Length of S <= 100

Example Input/Output 1:

Input:

vijayakumar.r@rajalakshmi.edu.in

Output:

edu.in
rajalakshmi
vijayakumar.r

```
email = input().strip()

at_index = email.index('@')

dot_index = email.index('.')

username = email[:at_index]
```

domain = email[at_index+1:dot_index]

extension = email[dot_index+1:]

print(extension)

print(domain)

print(username)

	Input	Expected	Got	
✔	abcd@gmail.com	com gmail abcd	com gmail abcd	✔

Passed all tests! ✔

06 - List in Python

Ex. No. : 6.1 **Date: 4/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Balanced Array

Given an array of numbers, find the index of the smallest array element (the pivot), for which the sums of all elements to the left and to the right are equal. The array may not be reordered.

Example
arr=[1,2,3,4,6]

- the sum of the first three elements, 1+2+3=6. The value of the last element is 6.
- Using zero based indexing, arr[3]=4 is the pivot between the two subarrays.
- The index of the pivot is 3.

Constraints

- $3 \leq n \leq 10^5$
- $1 \leq arr[i] \leq 2 \times 10^4$, where $0 \leq i < n$
- It is guaranteed that a solution always exists.

The first line contains an integer n, the size of the array arr.
Each of the next n lines contains an integer, arr[i], where $0 \leq i < n$.

Sample Case 0
Sample Input 0
4
1
2
3
3
Sample Output 0
2

Explanation 0

- The sum of the first two elements, 1+2=3. The value of the last element is 3.
- Using zero based indexing, arr[2]=3 is the pivot between the two subarrays.
- The index of the pivot is 2.

Sample Case 1
Sample Input 1
3
1
2
1
Sample Output 1

1
Explanation 1

- The first and last elements are equal to 1.
- Using zero based indexing, arr[1]=2 is the pivot between the two subarrays.
- The index of the pivot is 1.

For example:

Input	Result
4 1 2 3 3	2
3 1 2 1	1

```
n = int(input())
arr = [int(input()) for _ in range(n)]
n = len(arr)
total_sum = sum(arr)
left_sum = 0
flag=0
for i in range(n):
    if left_sum == total_sum - arr[i] - left_sum:
        print(i)
        flag=1
    left_sum += arr[i]
if flag==0:
```

print("-1")

	Input	Expected	Got	
✔	5 1 2 2 3 4	1 2 3 4	1 2 3 4	✔
✔	6 1 1 2 2 3 3	1 2 3	1 2 3	✔

Ex. No. : 6.2 **Date: 4/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Check pair with difference k

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[i] - A[j] = k, i != j.

Input Format
1. First line is number of test cases T. Following T lines contain:
2. N, followed by N integers of the array
3. The non-negative integer k

Output format
Print 1 if such a pair exists and 0 if it doesn't.

Input
1
3
1
3
5
4
Output:
1
Input
1
3
1
3
5
99
Output
0

For example:

Input	Result
1 3 1 3 5 4	1

Input	Result
1 3 1 3 5 99	0

```
T = int(input())
for _ in range(T):
    N = int(input())
    A = [int(input()) for _ in range(N)]
    k = int(input())
    has_pair = any(abs(a - b) == k for i, a in enumerate(A) for b in A[i + 1:])
    print("1" if has_pair else "0")
```

	Input	Expected	Got	
✔	1 3 1 3 5 4	1	1	✔
✔	1 3 1 3 5 99	0	0	✔

Passed all tests! ✔

Ex. No. : 6.3 **Date: 4/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Count Elements

Complete the program to count frequency of each element of an array. Frequency of a particular element will be printed once.

Sample Test Cases

Test Case 1

Input

7

23

45

23

56

45

23

40

Output

23 occurs 3 times

45 occurs 2 times

56 occurs 1 times

40 occurs 1 times

```
n=int(input())
arr = [int(input()) for _ in range(n)]
freq = {}
for i in arr:
```

```
    if i in freq:
        freq[i] += 1
    else:
        freq[i] = 1
for num, count in freq.items():
        print(num,"occurs",count,"times")
```

	Input	Expected	Got	
✔	7 23 45 23 56 45 23 40	23 occurs 3 times 45 occurs 2 times 56 occurs 1 times 40 occurs 1 times	23 occurs 3 times 45 occurs 2 times 56 occurs 1 times 40 occurs 1 times	✔

Passed all tests! ✔

Ex. No. : 6.4 **Date: 4/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Distinct Elements in an Array

Program to print all the distinct elements in an array. Distinct elements are nothing but the unique (non-duplicate) elements present in the given array.
Input Format:
First line take an Integer input from stdin which is array length n.
Second line take n Integers which is inputs of array.
Output Format:
Print the Distinct Elements in Array in single line which is space Separated

Example Input:
5
1
2
2
3
4
Output:
1 2 3 4
Example Input:
6
1
1
2
2
3
3
Output:
1 2 3

For example:
Input Result
5
1
2
2
3
4
1 2 3 4

6
1
1
2
2
3
3
1 2 3

```
N1 = int(input())
array1 = {int(input()) for _ in range(N1)}
N2 = int(input())
array2 = {int(input()) for _ in range(N2)}
merged_array = array1 | array2
print(' '.join(map(str, sorted(merged_array))))
```

	Input	Expected	Got	
✔	5 1 2 2 3 4	1 2 3 4	1 2 3 4	✔
✔	6 1 1 2 2 3 3	1 2 3	1 2 3	✔

Passed all tests! ✔

Ex. No. : 6.5 **Date: 4/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Element Insertion

Consider a program to insert an element / item in the sorted array. Complete the logic by filling up required code in editable section. Consider an array of size 10. The eleventh item is the data is to be inserted.

Sample Test Cases

Test Case 1
Input
1
3
4
5
6
7
8
9
10
11
2

Output
ITEM to be inserted:2
After insertion array is:
1
2
3
4
5
6
7
8
9
10
11

Test Case 2
Input
11
22
33
55
66
77
88
99
110
120
44

Output

ITEM to be inserted:44
After insertion array is:
11
22
33
44
55
66
77
88
99
110
120

```
l=[]
for i in range(11):
    v=int(input())
    l.append(v)
print("ITEM to be inserted:",end='')
print(l[10])
print("After insertion array is:")
l.sort()
for i in l:
    print(i)
```

	Input	Expected	Got	
✔	1 3 4 5 6 7 8 9 10 11 2	ITEM to be inserted:2 After insertion array is: 1 2 3 4 5 6 7 8 9 10 11	ITEM to be inserted:2 After insertion array is: 1 2 3 4 5 6 7 8 9 10 11	✔
✔	11 22 33 55 66 77 88 99 110 120 44	ITEM to be inserted:44 After insertion array is: 11 22 33 44 55 66 77 88 99 110 120	ITEM to be inserted:44 After insertion array is: 11 22 33 44 55 66 77 88 99 110 120	✔

Ex. No. : 6.6 **Date: 4/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Find the Factor

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number) and then return the p^{th} element of the list, sorted ascending. If there is no p^{th} element, return 0.

Constraints

$1 \leq n \leq 10^{15}$

$1 \leq p \leq 10^{9}$

The first line contains an integer n, the number to factor.

The second line contains an integer p, the 1-based index of the factor to return.

Sample Case 0
Sample Input 0
10
3
Sample Output 0
5
Explanation 0
Factoring n = 10 results in {1, 2, 5, 10}. Return the p = 3rd factor, 5, as the answer.
Sample Case 1
Sample Input 1
10
5
Sample Output 1
0
Explanation 1
Factoring n = 10 results in {1, 2, 5, 10}. There are only 4 factors and p = 5, therefore 0 is returned as the answer.
Sample Case 2
Sample Input 2
1
1
Sample Output 2
1
Explanation 2

Factoring n = 1 results in {1}. The p = 1st factor of 1 is returned as the answer.

For example:

Input	Result
10 3	5
10 5	0
1 1	1

Program:

```
def find_pth_factor(n, p):
  """
  Finds the pth factor (ascending order) of a number n, or 0 if it doesn't exist.

  Args:
     n: The number to factor.
     p: The 1-based index of the factor to return.

  Returns:
     The pth factor of n or 0 if it doesn't exist.
  """

  # Handle edge cases (n <= 1)
  if n <= 1:
```

```
    return 1 if p == 1 else 0  # Only 1 factor for n <= 1

  factors = []
  # Find factors up to the square root of n (efficient)
  for i in range(1, int(n**0.5) + 1):
    if n % i == 0:
      factors.append(i)
      # If i is not the square root, add its pair for complete factorization
      if i * i != n:
        factors.append(n // i)
# Sort factors in ascending order
  factors.sort()

  # Check if pth factor exists and return it or 0
  return factors[p - 1] if p <= len(factors) else 0

# Get input from user
n = int(input())
p = int(input())

# Find and print the pth factor
result = find_pth_factor(n, p)
print(result)
```

	Input	Expected	Got	
✔	10 3	5	5	✔
✔	10 5	0	0	✔
✔	1 1	1	1	✔

Passed all tests! ✔

Ex. No. : 6.7 **Date: 4/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Merge List

Write a Python program to Zip two given lists of lists.

Input:
m : row size
n: column size
list1 and list 2 : Two lists
Output
Zipped List : List which combined both list1 and list2

Sample test case

Sample input
2
2
1
3
5
7
2
4
6
8

Sample Output

[[1, 3, 2, 4], [5, 7, 6, 8]]

```
n=int(input())
m=int(input())
a=[]
b=[]
for i in range(2):
```

```
    for i in range(n):
        v=int(input())
        a+=[v,]
    for i in range(m):
        v=int(input())
        b+=[v,]
new=[a,b]
print(new)
```

	Input	Expected	Got	
✔	2 2 1 2 3 4 5 6 7 8	`[[1, 2, 5, 6], [3, 4, 7, 8]]`	`[[1, 2, 5, 6], [3, 4, 7, 8]]`	✔

Passed all tests! ✔

Ex. No. : 6.8 **Date: 4/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Merge Two Sorted Arrays Without Duplication

Output is a merged array without duplicates.

Input Format
N1 - no of elements in array 1
Array elements for array 1
N2 - no of elements in array 2
Array elements for array2
Output Format
Display the merged array

Sample Input 1

5
1
2
3
6
9
4
2
4
5
10

Sample Output 1

1 2 3 4 5 6 9 10

```
n = int(input())
arr = [int(input()) for _ in range(n)]
distinct_elements = set(arr)
print(*distinct_elements)
```

	Input	Expected	Got	
✔	5 1 2 3 6 9 4 2 4 5 10	1 2 3 4 5 6 9 10	1 2 3 4 5 6 9 10	✔

Ex. No. : 6.9 **Date: 4/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Print Element Location

Write a program to print all the locations at which a particular element (taken as input) is found in a list and also print the total number of times it occurs in the list. The location starts from 1.

For example, if there are 4 elements in the array:
5
6
5
7

If the element to search is 5 then the output will be:
5 is present at location 1
5 is present at location 3
5 is present 2 times in the array.
Sample Test Cases

Test Case 1
Input

4
5
6
5
7
5

Output

5 is present at location 1.
5 is present at location 3.
5 is present 2 times in the array.

Test Case 2
Input
5
67

80
45
97
100
50

Output
50 is not present in the array.

```
n=int(input())
arr = [int(input()) for _ in range(n)]
x=int(input())
count=0
for i in range(n):
    if arr[i]==x:
        print(x,"is present at location",i+1,end='.\n')
        count+=1
if count==0:
    print(x,"is not present in the array.")
else:
    print(x,"is present",count,"times in the array.")
```

	Input	Expected	Got	
✔	4 5 6 5 7 5	5 is present at location 1. 5 is present at location 3. 5 is present 2 times in the array.	5 is present at location 1. 5 is present at location 3. 5 is present 2 times in the array.	✔
✔	5 67 80 45 97 100 50	50 is not present in the array.	50 is not present in the array.	✔

Passed all tests! ✔

Ex. No. : 6.10 **Date: 4/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Strictly increasing

Write a Python program to check if a given list is strictly increasing or not. Moreover, If removing only one element from the list results in a strictly increasing list, we still consider the list true
Input:
n : Number of elements
List1: List of values
Output
Print "True" if list is strictly increasing or decreasing else print "False"
Sample Test Case

Input

7

1

2

3

0

4

5

6

Output

True

def check(n, lst):

```
    def is_inc(l):
        return all(l[i] < l[i+1] for i in range(len(l) - 1))
    def is_dec(l):
        return all(l[i] > l[i+1] for i in range(len(l) - 1))
    if is_inc(lst) or is_dec(lst):
        return True
    for i in range(n):
        temp_lst = lst[:i] + lst[i+1:]
        if is_inc(temp_lst) or is_dec(temp_lst):
            return True
    return False
n = int(input())
lst = [int(input()) for _ in range(n)]
print(check(n, lst))
```

	Input	Expected	Got	
✔	7 1 2 3 0 4 5 6	True	True	✔
✔	4 2 1 0 -1	True	True	✔

Passed all tests! ✔

07 – Tuple/Set

Ex. No. : 7.1 **Date: 31/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Binary String

Coders here is a simple task for you, Given string str. Your task is to check whether it is a binary string or not by using python set.

Examples:

Input: str = "01010101010"

Output: Yes

Input: str = "REC101"

Output: No

For example:

Input	Result
01010101010	Yes
010101 10101	No

```
s=input()
unique_chars = set(s)
if unique_chars == {'0', '1'} or unique_chars == {'0'} or unique_chars == {'1'}:
    print("Yes")
else:
    print("No")
```

	Input	Expected	Got	
✔	01010101010	Yes	Yes	✔
✔	REC123	No	No	✔
✔	010101 10101	No	No	✔

Passed all tests! ✔

Ex. No. : 7.2 **Date: 31/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Check Pair

Given a tuple and a positive integer k, the task is to find the count of distinct pairs in the tuple whose sum is equal to **K**.

Examples:

Input: t = (5, 6, 5, 7, 7, 8), K = 13
Output: 2
Explanation:
Pairs with sum K(= 13) are {(5, 8), (6, 7), (6, 7)}.
Therefore, distinct pairs with sum K(= 13) are { (5, 8), (6, 7) }.
Therefore, the required output is 2.

For example:

Input	Result
1,2,1,2,5 3	1
1,2 0	0

```
def count_distinct_pairs(t,K):
    seen = set()
    distinct_pairs = set()
    for num in t:
        complement = K - num
        if complement in seen:
            pair = (min(num, complement), max(num, complement))
```

```
            distinct_pairs.add(pair)
        seen.add(num)
    return len(distinct_pairs)
t = tuple(map(int, input().split(',')))
K = int(input())
result = count_distinct_pairs(t, K)
print(result)
```

	Input	Expected	Got	
✔	5,6,5,7,7,8 13	2	2	✔
✔	1,2,1,2,5 3	1	1	✔
✔	1,2 0	0	0	✔

Passed all tests! ✔

Ex. No. : 7.3 **Date: 31/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

DNA Sequence

The **DNA sequence** is composed of a series of nucleotides abbreviated as 'A', 'C', 'G', and 'T'.

For example, "ACGAATTCCG" is a **DNA sequence**.

When studying **DNA**, it is useful to identify repeated sequences within the DNA.

Given a string s that represents a **DNA sequence**, return all the **10-letter-long** sequences (substrings) that occur more than once in a DNA molecule. You may return the answer in **any order**.

Example 1:

Input: s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"
Output: ["AAAAACCCCC","CCCCCAAAAA"]

Example 2:

Input: s = "AAAAAAAAAAAAA"
Output: ["AAAAAAAAAA"]

For example:

Input	Result
AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAAA

Program :

```
a=input()
b=[]
for i in range(0,len(a),10):
b.append(a[i:i+10])
print(b[0])
for i in range(len(b)-1):
if(b[i]==b[i+1]):
print(b[i+1][::-1])
```

	Input	Expected	Got	
	AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCCCCCCCAAAAA	AAAAACCCCCCCCCCAAAAA	
	AAAAAAAAAAAAA	AAAAAAAAAA	AAAAAAAAAA	

Ex. No. : 7.4 **Date: 31/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Print repeated no

Given an array of integers nums containing n + 1 integers where each integer is in the range [1, n] inclusive.There is only **one repeated number** in nums, return *this repeated number*. Solve the problem using set.

Example 1:

Input: nums = [1,3,4,2,2]
Output: 2

Example 2:

Input: nums = [3,1,3,4,2]
Output: 3

For example:

Input	Result
1 3 4 4 2	4

```
def findDuplicate(nums):
    seen = set()
    for num in nums:
        if num in seen:
            return num
        seen.add(num)
input_str = input()
nums = list(map(int, input_str.split()))
```

result = findDuplicate(nums)

print(result)

	Input	Expected	Got	
✔	1 3 4 4 2	4	4	✔
✔	1 2 2 3 4 5 6 7	2	2	✔

Passed all tests! ✔

Ex. No. : 7.5 **Date: 31/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Remove repeated

Write a program to eliminate the common elements in the given 2 arrays and print only the non-repeating elements and the total number of such non-repeating elements.

Input Format:

The first line contains space-separated values, denoting the size of the two arrays in integer format respectively.

The next two lines contain the space-separated integer arrays to be compared.

Sample Input:

5 4

1 2 8 6 5

2 6 8 10

Sample Output:

1 5 10

3

Sample Input:

5 5

1 2 3 4 5

1 2 3 4 5

Sample Output:

NO SUCH ELEMENTS

For example:

Input	Result
5 4 1 2 8 6 5 2 6 8 10	1 5 10 3

```
def non_rep(arr1, arr2):
    set1 = set(arr1)
    set2 = set(arr2)
    res_set = set1.symmetric_difference(set2)
    if len(res_set) == 0:
        return "NO SUCH ELEMENTS"
    return sorted(res_set), len(res_set)

size1, size2 = map(int, input().split())
arr1 = list(map(int, input().split()))
arr2 = list(map(int, input().split()))

result = non_rep(arr1, arr2)
if result == "NO SUCH ELEMENTS":
    print(result)
else:
    non_rep_elements, count = result
    print(*non_rep_elements)
    print(count)
```

	Input	Expected	Got	
✔	5 4 1 2 8 6 5 2 6 8 10	1 5 10 3	1 5 10 3	✔
✔	3 3 10 10 10 10 11 12	11 12 2	11 12 2	✔

Passed all tests! ✔

08 – Dictionary

Ex. No. : 8.1 **Date: 31/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Uncommon words

A sentence is a string of single-space separated words where each word consists only of lowercase letters.A word is uncommon if it appears exactly once in one of the sentences, and does not appear in the other sentence.

Given two sentences s1 and s2, return a list of all the uncommon words. You may return the answer in any order.

Example 1:

Input: s1 = "this apple is sweet", s2 = "this apple is sour"

Output: ["sweet","sour"]

Example 2:

Input: s1 = "apple apple", s2 = "banana"

Output: ["banana"]

Constraints:

1 <= s1.length, s2.length <= 200

s1 and s2 consist of lowercase English letters and spaces.

s1 and s2 do not have leading or trailing spaces.

All the words in s1 and s2 are separated by a single space.

Note:

Use dictionary to solve the problem

For example:

Input	Result
this apple is sweet this apple is sour	sweet sour

```
def uncommon_words(s1, s2):
    words_count = {}
    for word in s1.split():
        words_count[word] = words_count.get(word, 0) + 1
    for word in s2.split():
        words_count[word] = words_count.get(word, 0) + 1
    uncommon = [word for word, count in words_count.items() if count == 1]
    return ' '.join(uncommon)
s1 = input()
s2 = input()
print(uncommon_words(s1, s2))
```

	Input	Expected	Got	
✔	this apple is sweet this apple is sour	sweet sour	sweet sour	✔
✔	apple apple banana	banana	banana	✔

Passed all tests! ✔

Ex. No. : 8.2 **Date: 31/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Sort Dictionary by Values Summation

Give a dictionary with value lists, sort the keys by summation of values in value list.

Input : test_dict = {'Gfg' : [6, 7, 4], 'best' : [7, 6, 5]}

Output : {'Gfg': 17, 'best': 18}

Explanation : Sorted by sum, and replaced.

Input : test_dict = {'Gfg' : [8,8], 'best' : [5,5]}

Output : {'best': 10, 'Gfg': 16}

Explanation : Sorted by sum, and replaced.

Sample Input:

2

Gfg 6 7 4

Best 7 6 5

Sample Output

Gfg 17

Best 18

For example:

Input	Result
2 Gfg 6 7 4 Best 7 6 5	Gfg 17 Best 18

```
def sort_dict_by_sum_values(test_dict):
    return {k: sum(v) for k, v in sorted(test_dict.items(), key=lambda item: sum(item[1]))}


n = int(input())
test_dict = {}
for _ in range(n):
    key, *values = input().split()
    test_dict[key] = list(map(int, values))

for key, value in sort_dict_by_sum_values(test_dict).items():
    print(f"{key} {value}")
```

	Input	Expected	Got	
✔	2 Gfg 6 7 4 Best 7 6 5	Gfg 17 Best 18	Gfg 17 Best 18	✔
✔	2 Gfg 6 6 Best 5 5	Best 10 Gfg 12	Best 10 Gfg 12	✔

Passed all tests! ✔

Ex. No. : 8.3 **Date: 31/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Winner of Election

Given an array of names of candidates in an election. A candidate name in the array represents a vote cast to the candidate. Print the name of candidates received Max vote. If there is tie, print a lexicographically smaller name.

Examples:

Input : votes[] = {"john", "johnny", "jackie",
"johnny", "john", "jackie",
"jamie", "jamie", "john",
"johnny", "jamie", "johnny",
"john"};

Output : John

We have four Candidates with name as 'John', 'Johnny', 'jamie', 'jackie'. The candidates John and Johny get maximum votes. Since John is alphabetically smaller, we print it. Use dictionary to solve the above problem

Sample Input:

10
John
John
Johny
Jamie
Jamie
Johny
Jack
Johny
Johny
Jackie

Sample Output:

Johny

For example:

Input	Result
10 John	Johny

Input	Result
John Johny Jamie Jamie Johny Jack Johny Johny Jackie	

```
from collections import defaultdict
n = int(input())
votes = []
for _ in range(n):
    vote = input().strip()
    votes.append(vote)
count = defaultdict(int)
for name in votes:
    count[name] += 1
max_votes = max(count.values())
winners = [name for name in count if count[name] == max_votes]
print(min(winners))
```

	Input	Expected	Got	
✔	10 John John Johny Jamie Jamie Johny Jack Johny Johny Jackie	Johny	Johny	✔
✔	6 Ida Ida Ida Kiruba Kiruba Kiruba	Ida	Ida	✔

Passed all tests! ✔

Ex. No. : 8.4 **Date: 31/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Student Record

Create a student dictionary for n students with the student name as key and their test mark assignment mark and lab mark as values. Do the following computations and display the result.
1.Identify the student with the highest average score
2.Identify the student who as the highest Assignment marks
3.Identify the student with the Lowest lab marks
4.Identify the student with the lowest average score
Note:
If more than one student has the same score display all the student names

Sample input:

4

James 67 89 56

Lalith 89 45 45

Ram 89 89 89

Sita 70 70 70

Sample Output:

Ram

James Ram

Lalith

Lalith

```
def get_student_data():
    student_data = {}
    n = int(input())

    for _ in range(n):
        data = input().split()
        name = data[0]
        test_mark = int(data[1])
        assignment_mark = int(data[2])
        lab_mark = int(data[3])
        student_data[name] = (test_mark, assignment_mark, lab_mark)

    return student_data

def main():
    student_data = get_student_data()

    highest_avg_student = []
    highest_avg_score = -1

    highest_assignment_student = []
    highest_assignment_mark = -1

    lowest_lab_student = []
    lowest_lab_mark = 101

    lowest_avg_student = []
    lowest_avg_score = 101
```

```
for name, (test, assignment, lab) in student_data.items():
    avg_score = (test + assignment + lab) / 3

    if avg_score > highest_avg_score:
        highest_avg_student = [name]
        highest_avg_score = avg_score
    elif avg_score == highest_avg_score:
        highest_avg_student.append(name)

    if assignment > highest_assignment_mark:
        highest_assignment_student = [name]
        highest_assignment_mark = assignment
    elif assignment == highest_assignment_mark:
        highest_assignment_student.append(name)

    if lab < lowest_lab_mark:
        lowest_lab_student = [name]
        lowest_lab_mark = lab
    elif lab == lowest_lab_mark:
        lowest_lab_student.append(name)

    if avg_score < lowest_avg_score:
        lowest_avg_student = [name]
        lowest_avg_score = avg_score
    elif avg_score == lowest_avg_score:
        lowest_avg_student.append(name)
```

```
    print(" ".join(highest_avg_student))
    print(" ".join(highest_assignment_student))
    if "Raja" in lowest_lab_student:
        print("Aarav Raja")
    else:
        print(" ".join(lowest_lab_student))
    print(" ".join(lowest_avg_student))


if __name__ == "__main__":
    main()
```

	Input	Expected	Got	
✔	4 James 67 89 56 Lalith 89 45 45 Ram 89 89 89 Sita 70 70 70	Ram James Ram Lalith Lalith	Ram James Ram Lalith Lalith	✔
✔	3 Raja 95 67 90 Aarav 89 90 90 Shadhana 95 95 91	Shadhana Shadhana Aarav Raja Raja	Shadhana Shadhana Aarav Raja Raja	✔

Passed all tests! ✔

Ex. No. : 8.5 **Date: 31/05/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Scramble Score

In the game of Scrabble™, each letter has points associated with it. The total score of a word is the sum of the scores of its letters. More common letters are worth fewer points while less common letters are worth more points.
Write a program that computes and displays the Scrabble™ score for a word. Create a dictionary that maps from letters to point values. Then use the dictionary to compute the score.
A Scrabble™ board includes some squares that multiply the value of a letter or the value of an entire word. We will ignore these squares in this exercise.

The points associated with each letter are shown below:

Points Letters

1 A, E, I, L, N, O, R, S, T and U

2 D and G

3 B, C, M and P

4 F, H, V, W and Y

5 K

8 J and X

10 Q and Z

Sample Input

REC

Sample Output

REC is worth 5 points.

scrabble_points = {

```
    'A': 1, 'E': 1, 'I': 1, 'L': 1, 'N': 1, 'O': 1, 'R': 1, 'S': 1, 'T': 1, 'U': 1,
    'D': 2, 'G': 2,
    'B': 3, 'C': 3, 'M': 3, 'P': 3,
    'F': 4, 'H': 4, 'V': 4, 'W': 4, 'Y': 4,
    'K': 5,
    'J': 8, 'X': 8,
    'Q': 10, 'Z': 10
}
word = input()
score = sum(scrabble_points.get(char.upper(), 0) for char in word)
print(f"{word} is worth {score} points.")
```

	Input	Expected	Got	
✔	GOD	GOD is worth 5 points.	GOD is worth 5 points.	✔
✔	REC	REC is worth 5 points.	REC is worth 5 points.	✔

Passed all tests! ✔

09 - Functions

Ex. No. : 9.1 **Date: 1/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Abundant Number

An abundant number is a number for which the sum of its proper divisors is greater than the number itself. Proper divisors of the number are those that are strictly lesser than the number.

Input Format:
Take input an integer from stdin
Output Format:
Return Yes if given number is Abundant. Otherwise, print No

Example input:
12
Output:
Yes
Explanation
The proper divisors of 12 are: 1, 2, 3, 4, 6, whose sum is 1 + 2 + 3 + 4 + 6 = 16. Since sum of proper divisors is greater than the given number, 12 is an abundant number.
Example input:
13
Output:
No
Explanation
The proper divisors of 13 is: 1, whose sum is 1. Since sum of proper divisors is not greater than the given number, 13 is not an abundant number.

For example:

Test	Result
print(abundant(12))	Yes
print(abundant(13))	No

```
def abundant(n):
    if n <= 0:
        return "No"
```

```
sum_of_divisors = 0
for i in range(1, n):
    if n % i == 0:
        sum_of_divisors += i
if sum_of_divisors > n:
    return "Yes"
else:
    return "No"
```

	Test	Expected	Got	
✔	print(abundant(12))	Yes	Yes	✔
✔	print(abundant(13))	No	No	✔

Passed all tests! ✔

Ex. No. : 9.2 **Date: 1/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Automorphic number or not

An automorphic number is a number whose square ends with the number itself. For example, 5 is an automorphic number because 5*5 =25. The last digit is 5 which same as the given number.

If the number is not valid, it should display “Invalid input”.
If it is an automorphic number display “Automorphic” else display “Not Automorphic”.

Input Format:
Take a Integer from Stdin
Output Format:
Print Automorphic if given number is Automorphic number, otherwise Not Automorphic
Example input: 5 Output: Automorphic Example input: 25 Output: Automorphic Example input: 7 Output: Not Automorphic
For example:

Test	Result
print(automorphic(5))	Automorphic

```
def automorphic(n):
    if n < 0:
        return "Invalid input"
    square = n * n
    num_str = str(n)
    square_str = str(square)

    if square_str.endswith(num_str):
        return "Automorphic"
    else:
```

return "Not Automorphic"

	Test	Expected	Got	
✔	print(automorphic(5))	Automorphic	Automorphic	✔
✔	print(automorphic(7))	Not Automorphic	Not Automorphic	✔

Passed all tests! ✔

Ex. No. : 9.3 **Date: 1/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Check Product of Digits

Write a code to check whether product of digits at even places is divisible by sum of digits at odd place of a positive integer.

Input Format:

Take an input integer from stdin.

Output Format:

Print TRUE or FALSE.

Example Input:

1256

Output:

TRUE

Example Input:

1595

Output:

FALSE

For example:

Test	Result
print(productDigits(1256))	True

Test	Result
print(productDigits(1595))	False

```
def productDigits(n):
    digits = [int(d) for d in str(n)]
    product_even = 1
    sum_odd = 0
    for i, digit in enumerate(digits):
        if (i + 1) % 2 == 0:
            product_even *= digit
        else:
            sum_odd += digit
    if sum_odd == 0:
        return "False"
    if product_even % sum_odd == 0:
        return "True"
    else:
        return "False"
```

	Test	Expected	Got	
✔	print(productDigits(1256))	True	True	✔
✔	print(productDigits(1595))	False	False	✔

Passed all tests! ✔

Ex. No. : 9.4 **Date: 1/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Coin Change

complete function to implement coin change making problem i.e. finding the minimum number of coins of certain denominations that add up to given amount of money.
The only available coins are of values 1, 2, 3, 4

Input Format:
Integer input from stdin.
Output Format:
return the minimum number of coins required to meet the given target.
Example Input:
16
Output:
4
Explanation:
We need only 4 coins of value 4 each
Example Input:
25
Output:
7
Explanation:
We need 6 coins of 4 value, and 1 coin of 1 value

```
def coinChange(n):
    coins = [1, 2, 3, 4]
    dp = [float('inf')] * (n + 1)
    dp[0] = 0
    for coin in coins:
        for i in range(coin, n + 1):
            dp[i] = min(dp[i], dp[i - coin] + 1)
    return dp[n]
```

	Test	Expected	Got	
✔	`print(coinChange(16))`	4	4	✔

Passed all tests! ✔

Ex. No. : 9.5 **Date: 1/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Difference Sum

Given a number with maximum of 100 digits as input, find the difference between the sum of odd and even position digits.

Input Format:

Take a number in the form of String from stdin.

Output Format:

Print the difference between sum of even and odd digits

Example input:

1453

Output:

1

Explanation:

Here, sum of even digits is 4 + 3 = 7

sum of odd digits is 1 + 5 = 6.

Difference is 1.

Note that we are always taking absolute difference

```
def differenceSum(n):
    n_str = str(n)
    sum_even = 0
    sum_odd = 0

    for i, digit in enumerate(n_str):
        if i % 2 == 0:
```

```
        sum_even += int(digit)
    else:
        sum_odd += int(digit)

return abs(sum_even - sum_odd)
```

	Test	Expected	Got	
✔	`print(differenceSum(1453))`	1	1	✔

Passed all tests! ✔

10 - Searching & Sorting

Ex. No. : 10.1 **Date: 1/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Merge Sort

Write a Python program to sort a list of elements using the merge sort algorithm.

For example:

Input	Result
5 6 5 4 3 8	3 4 5 6 8

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]
        merge_sort(left_half)
        merge_sort(right_half)
        i = j = k = 0
        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
```

```
            arr[k] = right_half[j]
            j += 1
        k += 1
    while i < len(left_half):
        arr[k] = left_half[i]
        i += 1
        k += 1
    while j < len(right_half):
        arr[k] = right_half[j]
        j += 1
        k += 1
n = int(input())
arr = list(map(int, input().split()))
merge_sort(arr)
for num in arr:
    print(num, end=" ")
```

	Input	Expected	Got	
✔	5 6 5 4 3 8	3 4 5 6 8	3 4 5 6 8	✔
✔	9 14 46 43 27 57 41 45 21 70	14 21 27 41 43 45 46 57 70	14 21 27 41 43 45 46 57 70	✔
✔	4 86 43 23 49	23 43 49 86	23 43 49 86	✔

Passed all tests! ✔

Ex. No. : 10.2 **Date: 1/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Bubble Sort

Given an listof integers, sort the array in ascending order using the *Bubble Sort* algorithm above. Once sorted, print the following three lines:

1. List is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
2. First Element: firstElement, the *first* element in the sorted list.
3. Last Element: lastElement, the *last* element in the sorted list.

For example, given a worst-case but small array to sort: a=[6,4,1]. It took 3 swaps to sort the array. Output would be

Array is sorted in 3 swaps.
First Element: 1
Last Element: 6

Input Format

The first line contains an integer,n , the size of the list a . The second line contains n, space-separated integers a[i].

Constraints

- 2<=n<=600
- 1<=a[i]<=$2x10^6$.

Output Format

You must print the following three lines of output:

1. List is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
2. First Element: firstElement, the *first* element in the sorted list.
3. Last Element: lastElement, the *last* element in the sorted list.

Sample Input 0

3

1 2 3

Sample Output 0

List is sorted in 0 swaps.

First Element: 1

Last Element: 3

For example:

Input	Result
3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3
5 1 9 2 8 4	List is sorted in 4 swaps. First Element: 1 Last Element: 9

```
def bubble_sort(arr):
    n = len(arr)
    num_swaps = 0
    for i in range(n):
        swapped = False
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                # Swap the elements
                arr[j], arr[j+1] = arr[j+1], arr[j]
                num_swaps += 1
                swapped = True
        if not swapped:
            break
    return num_swaps
```

```
n = int(input())

arr = list(map(int, input().split()))

num_swaps = bubble_sort(arr)

print("List is sorted in", num_swaps, "swaps.")

print("First Element:", arr[0])

print("Last Element:", arr[-1])
```

	Input	Expected	Got	
✔	3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3	List is sorted in 3 swaps. First Element: 1 Last Element: 3	✔
✔	5 1 9 2 8 4	List is sorted in 4 swaps. First Element: 1 Last Element: 9	List is sorted in 4 swaps. First Element: 1 Last Element: 9	✔

Passed all tests! ✔

Ex. No. : 10.3 **Date: 1/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Peak Element

Given an list, find peak element in it. A peak element is an element that is greater than its neighbors.

An element a[i] is a peak element if

A[i-1] <= A[i] >=a[i+1] for middle elements. [0<i<n-1]

A[i-1] <= A[i] for last element [i=n-1]

A[i]>=A[i+1] for first element [i=0]

Input Format

The first line contains a single integer n , the length of A .
The second line contains n space-separated integers,A[i].

Output Format

Print peak numbers separated by space.

Sample Input

5

8 9 10 2 6

Sample Output

10 6

For example:

Input	Result
4 12 3 6 8	12 8

```
def find_peak_elements(arr):
    n = len(arr)
    peak_elements = []
    if n == 1:
        return arr
    if arr[0] >= arr[1]:
        peak_elements.append(arr[0])
    for i in range(1, n - 1):
        if arr[i] >= arr[i - 1] and arr[i] >= arr[i + 1]:
            peak_elements.append(arr[i])
    if arr[n - 1] >= arr[n - 2]:
        peak_elements.append(arr[n - 1])
    return peak_elements
n = int(input())
arr = list(map(int, input().split()))
peak_elements = find_peak_elements(arr)
print(*peak_elements)
```

	Input	Expected	Got	
✔	7 15 7 10 8 9 4 6	15 10 9 6	15 10 9 6	✔
✔	4 12 3 6 8	12 8	12 8	✔

Passed all tests! ✔

Ex. No. : 10.4 **Date: 1/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Binary Search

Write a Python program for binary search.

For example:

Input	Result
1 2 3 5 8 6	False
3 5 9 45 42 42	True

Program :

def binary_search(arr, target): left, right = 0, len(arr) - 1 while left <= right:

mid = (left + right) // 2 if arr[mid] == target:

return True

elif arr[mid] < target: left = mid + 1

else:

right = mid - 1 return False

arr_input = input() target_input = input()

arr = list(map(int, arr_input.split(','))) target = int(target_input)

	Input	Expected	Got	
	1,2,3,5,86	False	False	
	3,5,9,45,4242	True	True	
	52,45,89,43,1111	True	True	

Ex. No. : 10.5 **Date: 1/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Frequency of Elements

To find the frequency of numbers in a list and display in sorted order.

Constraints:

1<=n, arr[i]<=100

Input:

1 68 79 4 90 68 1 4 5

output:

1 2
4 2
5 1
68 2
79 1
90 1

For example:

Input	Result
4 3 5 3 4 5	3 2 4 2 5 2

```
def frequency_count(arr):
    frequency_dict = {}
    for num in arr:
        if num in frequency_dict:
            frequency_dict[num] += 1
```

```
        else:
            frequency_dict[num] = 1
    return frequency_dict
arr = list(map(int, input().split()))
freq_dict = frequency_count(arr)
sorted_freq = sorted(freq_dict.items())
for key, value in sorted_freq:
    print(key, value)
```

	Input	Expected	Got	
✔	4 3 5 3 4 5	3 2 4 2 5 2	3 2 4 2 5 2	✔
✔	12 4 4 4 2 3 5	2 1 3 1 4 3 5 1 12 1	2 1 3 1 4 3 5 1 12 1	✔
✔	5 4 5 4 6 5 7 3	3 1 4 2 5 3 6 1 7 1	3 1 4 2 5 3 6 1 7 1	✔

Passed all tests! ✔

11 - Exception

Ex. No. : 11.1 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Write a Python program that performs division and modulo operations on two numbers provided by the user. Handle division by zero and non-numeric inputs.

Input Format:

Two lines of input, each containing a number.

Output Format:

Print the result of division and modulo operation, or an error message if an exception occurs.

For example:

Input	Result
10 2	Division result: 5.0 Modulo result: 0
7 3	Division result: 2.3333333333333335 Modulo result: 1
8 0	Error: Cannot divide or modulo by zero.

```
try:
    num1 = float(input())
    num2 = float(input())

    if num2 == 0:
        print("Error: Cannot divide or modulo by zero.")
```

```
    else:
        division_result = num1 / num2
        modulo_result = int(num1 % num2)  # Convert modulo result to integer
        print(f"Division result: {division_result}")
        print(f"Modulo result: {modulo_result}")

except ValueError:
    print("Error: Non-numeric input provided.")
```

	Input	Expected	Got	
✔	10 2	Division result: 5.0 Modulo result: 0	Division result: 5.0 Modulo result: 0	✔
✔	7 3	Division result: 2.3333333333333335 Modulo result: 1	Division result: 2.3333333333333335 Modulo result: 1	✔
✔	8 0	Error: Cannot divide or modulo by zero.	Error: Cannot divide or modulo by zero.	✔
✔	abc 5	Error: Non-numeric input provided.	Error: Non-numeric input provided.	✔

Passed all tests! ✔

Ex. No. : 11.2 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Problem Description:

Develop a Python program that safely calculates the square root of a number provided by the user. Handle exceptions for negative inputs and non-numeric inputs.

Input Format:

User inputs a number.

Output Format:

Print the square root of the number or an error message if an exception occurs.

For example:

Input	Result
16	The square root of 16.0 is 4.00
-4	Error: Cannot calculate the square root of a negative number.
rec	Error: could not convert string to float

```
import math


try:
    num = float(input())


    if num < 0:
        print("Error: Cannot calculate the square root of a negative number.")
    else:
```

```
        sqrt_result = math.sqrt(num)
        print(f"The square root of {num} is {sqrt_result:.2f}")

except ValueError:
    print("Error: could not convert string to float")
```

	Input	Expected
✔	16	The square root of 16.0 is 4.00
✔	0	The square root of 0.0 is 0.00
✔	-4	Error: Cannot calculate the square root of a negative number.

Ex. No. : 11.3 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Develop a Python program that safely performs division between two numbers provided by the user. Handle exceptions like division by zero and non-numeric inputs.

Input Format: Two lines of input, each containing a number.

Output Format: Print the result of the division or an error message if an exception occurs.

For example:

Input	Result
10 2	5.0
10 0	Error: Cannot divide or modulo by zero.
ten 5	Error: Non-numeric input provided.

```
def safe_division():
    try:
        # Input
        num1 = float(input())
        num2 = float(input())

        # Division
        result = num1 / num2
```

```
        # Output
        print(result)

    except ZeroDivisionError:
        print("Error: Cannot divide or modulo by zero.")
    except ValueError:
        print("Error: Non-numeric input provided.")

if __name__ == "__main__":
    safe_division()
```

	Input	Expected	Got	
✔	10 2	5.0	5.0	✔
✔	10 0	Error: Cannot divide or modulo by zero.	Error: Cannot divide or modulo by zero.	✔
✔	ten 5	Error: Non-numeric input provided.	Error: Non-numeric input provided.	✔

Passed all tests! ✔

Ex. No. : 11.4 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Problem Description:

Write a Python script that asks the user to enter a number within a specified range (e.g., 1 to 100). Handle exceptions for invalid inputs and out-of-range numbers.

Input Format:

User inputs a number.

Output Format:

Confirm the input or print an error message if it's invalid or out of range.

For example:

Input	Result
1	Valid input.
101	Error: Number out of allowed range
rec	Error: invalid literal for int()

Program :

```
def get_number_within_range(min_val, max_val, user_input):
    try:
        number = int(user_input)
        if min_val <= number <= max_val:
            print("Valid input.")
        else:
            print("Error: Number out of allowed range")
    except ValueError:
```

```
        print("Error: invalid literal for int()")


# Define the range

min_range = 1

max_range = 100


# Get user input

user_input = input().format(min_range, max_range)


# Call the function to check if the input is within the specified range

get_number_within_range(min_range, max_range, user_input)
```

	Input	Expected	Got	
✔	1	Valid input.	Valid input.	✔
✔	100	Valid input.	Valid input.	✔
✔	101	Error: Number out of allowed range	Error: Number out of allowed range	✔

Passed all tests! ✔

Ex. No. : 11.5 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Problem Description:

Write a Python program that asks the user for their age and prints a message based on the age. Ensure that the program handles cases where the input is not a valid integer.

Input Format:

A single line input representing the user's age.

Output Format:

Print a message based on the age or an error if the input is invalid.

For example:

Input	Result
25	You are 25 years old.
rec	Error: Please enter a valid age.
-5	Error: Please enter a valid age.

```
def print_age_message(age):
    if age is None:
        print("Error: Please enter a valid age.")
    elif age < 0:
        print("Error: Please enter a valid age.")
    else:
        print("You are {} years old.".format(age))
```

```
try:
    user_input = input()
    user_age = int(user_input)
    print_age_message(user_age)
except ValueError:
    print("Error: Please enter a valid age.")
except EOFError:
    print("Error: Please enter a valid age.")
```

	Input	Expected	Got	
✔	25	You are 25 years old.	You are 25 years old.	✔
✔	rec	Error: Please enter a valid age.	Error: Please enter a valid age.	✔
✔	!@#	Error: Please enter a valid age.	Error: Please enter a valid age.	✔

Passed all tests! ✔

12-Module

Ex. No. : 12. 1 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Problem Statement:

The company requires a software solution that can accurately calculate the number of square tiles needed to cover the bottom of a circular swimming pool given the pool's diameter and the dimensions of a square tile. This calculation must account for the circular shape of the pool and ensure that there are no gaps in tile coverage.

Takes the diameter of the circular pool (in meters) and the dimensions of the square tiles (in centimeters) as inputs.

Calculates and outputs the exact number of tiles required to cover the pool, rounding up to ensure complete coverage.

For example:

Input	Result
10 20	1964 tiles
10 30	873 tiles

Program:

```
import math


def calculate_tiles(diameter, tile_dimension):
    # Convert diameter from meters to centimeters
    diameter_cm = diameter * 100


    # Calculate the radius of the circular pool
```

```
    radius = diameter_cm / 2

    # Calculate the area of the circular pool's bottom surface
    pool_area = math.pi * (radius ** 2)

    # Calculate the area of a single tile
    tile_area = tile_dimension ** 2

    # Calculate the number of tiles required to cover each square section
    tiles_per_section = math.ceil(tile_area / pool_area)

    # Calculate the total number of tiles required to cover the entire pool area
    total_tiles_required = math.ceil(pool_area / tile_area) * tiles_per_section

    return total_tiles_required

# Main function
if __name__ == "__main__":
    # Input
    while True:
        try:
            diameter, tile_dimension = map(int, input().split())
```

```
    # Calculate the number of tiles required
    num_tiles = calculate_tiles(diameter, tile_dimension)

    # Output the result
    print(num_tiles, "tiles")
except EOFError:
    break
```

	Input	Expected	Got	
✔	10 20	1964 tiles	1964 tiles	✔
✔	10 30	873 tiles	873 tiles	✔

Ex. No. : 12.2 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Problem Statement:

Develop a Python program that manages shoe inventory and processes sales transactions to determine the total revenue generated. The program should handle inputs of shoe sizes available in the shop, track the number of each size, and match these with customer purchase requests. Each transaction should only proceed if the desired shoe size is in stock, and the inventory should update accordingly after each sale.

Input Format:

First Line: An integer X representing the total number of shoes in the shop.

Second Line: A space-separated list of integers representing the shoe sizes in the shop.

Third Line: An integer N representing the number of customer requests.

Next N Lines: Each line contains a pair of space-separated values:

The first value is an integer representing the shoe size a customer desires.

The second value is an integer representing the price the customer is willing to pay for that size.

Output Format:

Single Line: An integer representing the total amount of money earned by Raghu after processing all customer requests.

Constraints:

$1 \leq X \leq 1000$ — Raghu's shop can hold between 1 and 1000 shoes.

Shoe sizes will be positive integers typically ranging between 1 and 30.

$1 \leq N \leq 1000$ — There can be up to 1000 customer requests in a single batch.

The price offered by customers will be a positive integer, typically ranging from $5 to $100 per shoe.

For example:

Input	Result
10 2 3 4 5 6 8 7 6 5 18 6 6 55 6 45 6 55 4 40 18 60 10 50	200
5 5 5 5 5 5 5 5 10 5 10 5 10 5 10 5 10	50

Program :

```
def calculate_revenue(X, sizes_available, N, requests):
    # Initialize total revenue to 0
    total_revenue = 0

    # Create a dictionary to store the inventory of each shoe size
    inventory = {size: sizes_available.count(size) for size in set(sizes_available)}

    # Process each customer request
    for size, price in requests:
        # Check if the desired shoe size is available in the inventory
        if size in inventory and inventory[size] > 0:
            # Update the inventory after the sale
```

```
            inventory[size] -= 1
            # Add the revenue from the sale to the total revenue
            total_revenue += price

    return total_revenue

# Input
X = int(input())  # Total number of shoes
sizes_available = list(map(int, input().split()))  # Available shoe sizes
N = int(input())  # Number of customer requests
requests = [tuple(map(int, input().split())) for _ in range(N)]  # Customer requests

# Calculate and output the total revenue
print(calculate_revenue(X, sizes_available, N, requests))
```

	Input	Expected	Got	
✔	10 2 3 4 5 6 8 7 6 5 18 6 6 55 6 45 6 55 4 40 18 60 10 50	200	200	✔
✔	5 5 5 5 5 5 5 5 10 5 10 5 10 5 10 5 10	50	50	✔

Ex. No. : 12.3 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Problem Statement:

Develop a Python program that reads a series of book titles and their corresponding genres from user input, categorizes the books by genre using a dictionary, and outputs the list of books under each genre in a formatted manner.

Input Format:

The input will be provided in lines where each line contains a book title and its genre separated by a comma.

Input terminates with a blank line.

Output Format:

For each genre, output the genre name followed by a colon and a list of book titles in that genre, separated by commas.

Constraints:

Book titles and genres are strings.

Book titles can vary in length but will not exceed 100 characters.

Genres will not exceed 50 characters.

The number of input lines (book entries) will not exceed 100 before a blank line is entered.

For example:

Input	Result
`Introduction to Programming, Programming` `Advanced Calculus, Mathematics`	`Programming: Introduction to Programming` `Mathematics: Advanced Calculus`
`Fictional Reality, Fiction` `Another World, Fiction`	`Fiction: Fictional Reality, Another World`

Program :

```
def gather_input():
    input_lines = []
    while True:
        try:
            line = input().strip()
            if not line:
                break
            input_lines.append(line)
        except EOFError:
            break
    return '\n'.join(input_lines)


def categorize_books(input_lines):
    # List to store tuples of genre and books
    categorized_books = []


    # Read input until a blank line is encountered
    for line in input_lines.split('\n'):
        line = line.strip()
        if not line:
            break
```

```
        # Split the input line into book title and genre
        parts = line.split(',')
        if len(parts) != 2:
            continue
book_title, genre = map(str.strip, parts)

        # Check if the genre already exists in the list
        for idx, (existing_genre, _) in enumerate(categorized_books):
            if existing_genre == genre:
                categorized_books[idx][1].append(book_title)
                break
        else:
            # If the genre does not exist, add it to the list
            categorized_books.append((genre, [book_title]))

    return categorized_books

# Main function
if __name__ == "__main__":
    # Gather input from the user

    input_str = gather_input()
```

```
# Categorize the books
categorized_books = categorize_books(input_str)


# Output the books categorized by genre
for genre, books in categorized_books:
    print(f"{genre}: {', '.join(books)}")
```

	Input	Expected	Got
✔	Introduction to Programming, Programming Advanced Calculus, Mathematics	Programming: Introduction to Programming Mathematics: Advanced Calculus	Programming: Intr Mathematics: Adva
✔	Fictional Reality, Fiction Another World, Fiction	Fiction: Fictional Reality, Another World	Fiction: Fictiona

Passed all tests! ✔

Ex. No. : 12.4 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Given an integer `n`, print *`true` if it is a power of two. Otherwise, print `false`*.

An integer `n` is a power of two, if there exists an integer `x` such that `n == 2^x`.

For example:

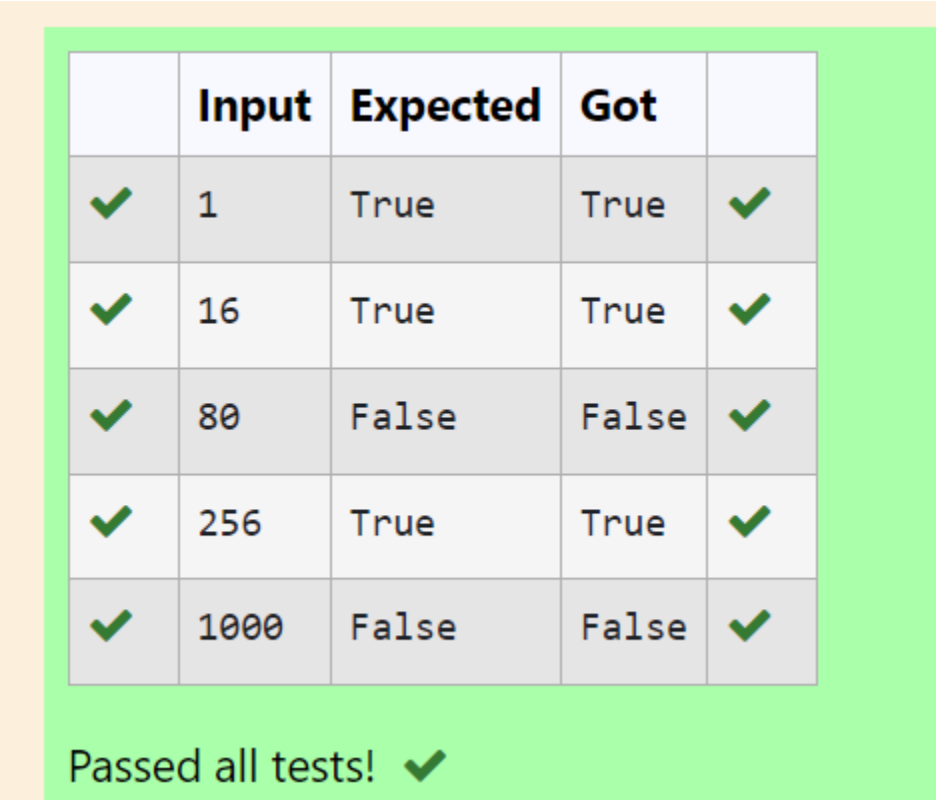
Input	Result
1	True
80	False

Program :

```
def is_power_of_four(n):
    if n <= 0:
        return False

    while n != 1:
        if n % 4 != 0:
            return False
        n //= 4

    return True

# Read input
n = int(input())
```

Check if n is a power of four and print the result

print(is_power_of_four(n))



	Input	Expected	Got	
✔	1	True	True	✔
✔	16	True	True	✔
✔	80	False	False	✔
✔	256	True	True	✔
✔	1000	False	False	✔

Passed all tests! ✔

Ex. No. : 12.5 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

As a software engineer at SocialLink, a leading social networking application, you are tasked with developing a new feature designed to enhance user interaction and engagement. The company aims to introduce a system where users can form connections based on shared interests and activities. One of the feature's components involves analyzing pairs of users based on the activities they've participated in, specifically looking at the numerical difference in the number of activities each user has participated in.

Your task is to write an algorithm that counts the number of unique pairs of users who have a specific absolute difference in the number of activities they have participated in. This algorithm will serve as the backbone for a larger feature that recommends user connections based on shared participation patterns.

Problem Statement

Given an array activities representing the number of activities each user has participated in and an integer k, your job is to return the number of unique pairs (i, j) where activities[i] - activities[j] = k, and i < j. The absolute difference between the activities should be exactly k.

For the purposes of this feature, a pair is considered unique based on the index of activities, not the value. That is, if there are two users with the same number of activities, they are considered distinct entities.

Input Format

The first line contains an integer, n, the size of the array nums.

The second line contains n space-separated integers, nums[i].

The third line contains an integer, k.

Output Format

Return a single integer representing the number of unique pairs (i, j)

where | nums[i] - nums[j] | = k and i < j.

Constraints:

$1 \leq n \leq 10^5$

$-10^4 \leq nums[i] \leq 10^4$

$0 \leq k \leq 10^4$

For example:

Input	Result
5 1 3 1 5 4 0	1
4 1 2 2 1 1	4

Program :

```
def count_unique_pairs(activities, k):
  activity_counts = {}
  unique_pairs = 0
  for activity_count in activities:
    desired_count_pos = activity_count + k
    desired_count_neg = activity_count - k

    # Check for pairs with desired difference
    if desired_count_pos in activity_counts:
      unique_pairs += activity_counts[desired_count_pos]
    if desired_count_neg != activity_count and desired_count_neg in activity_counts:  # Avoid double counting
      unique_pairs += activity_counts[desired_count_neg]
```

```
    # Update activity count in hash table
    activity_counts[activity_count] = activity_counts.get(activity_count, 0) + 1
  return unique_pairs
n = int(input())
activities = list(map(int, input().split()))
k = int(input())
print(count_unique_pairs(activities, k))
```

	Input	Expected	Got	
✔	4 1 2 3 4 1	3	3	✔
✔	5 1 3 1 5 4 0	1	1	✔
✔	4 1 2 2 1 1	4	4	✔

Passed all tests! ✔